

## Cybersecurity Professional Program

---

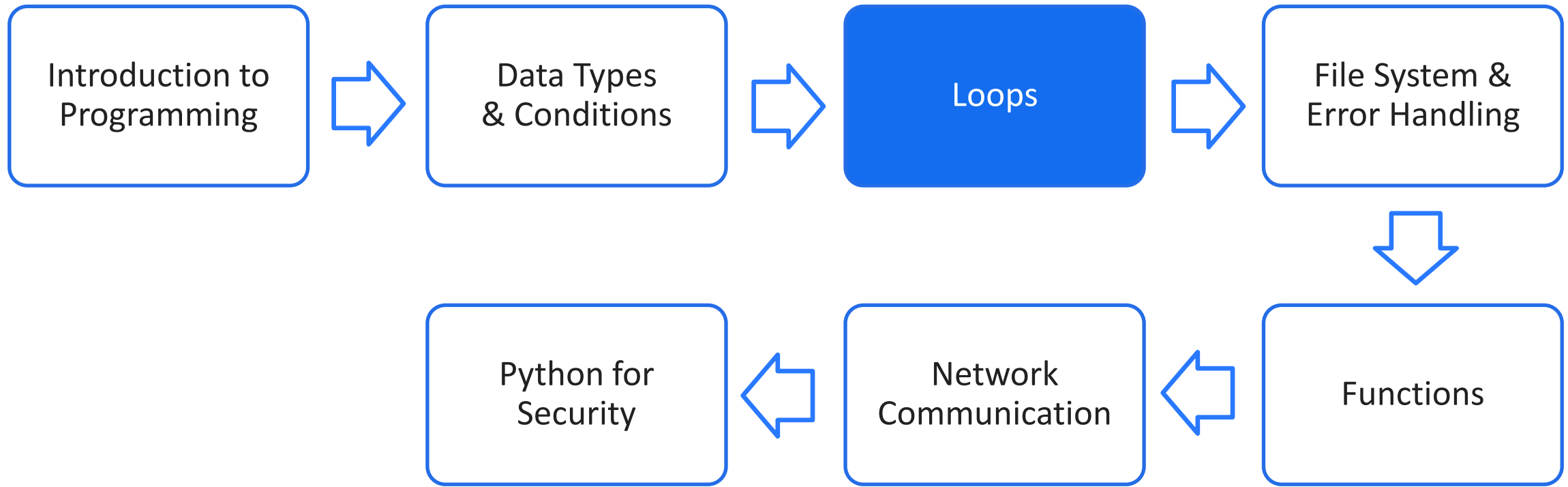
# Loops

Introduction to Python for Security



# Introduction to Python for Security

## Course Path



The graphic consists of three hexagons. The top-left hexagon is white with a blue outline. The bottom-left hexagon is white with a blue outline. The bottom-right hexagon is solid blue with a white target icon (a circle with an arrow in the center).

# Loops Objectives

This lesson is an introduction to Python ***for*** and ***while*** loops. During the lesson, management methods will be pointed out to emphasize organized and efficient code development.

- For & While Loops
- Loops & Conditions

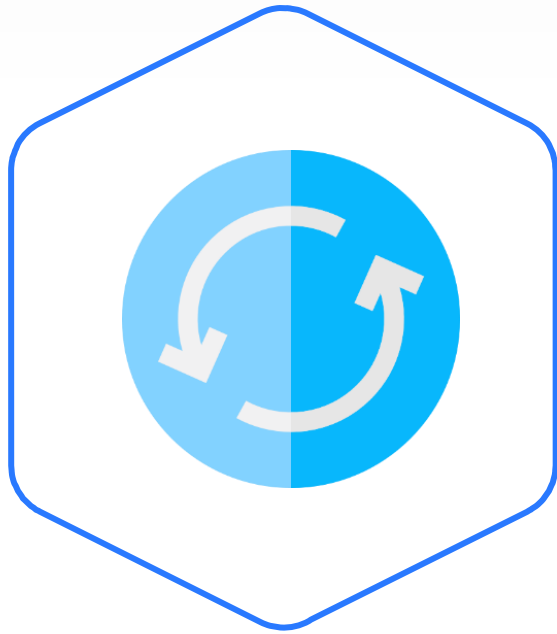


Loops

---

# For & While Loops

# What Are Loops?



- Loops are blocks of code that run as long as a condition is true.
- They allow repeating code logic for as long as the condition is true.



# For & While Loops

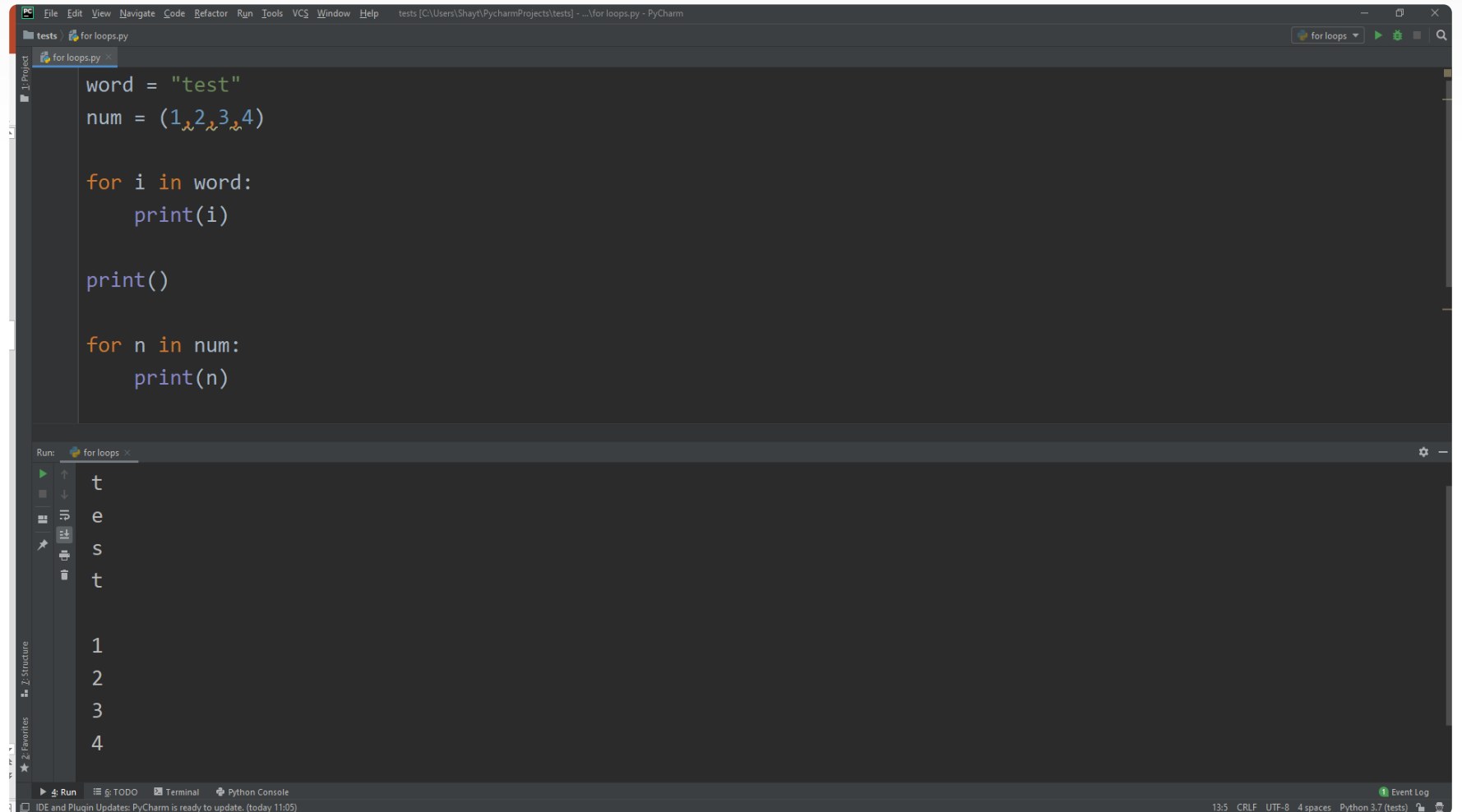
## For Loops



Perform blocks of code  
if the condition is true

A **for** loop provides  
iterating capabilities.

Can iterate over lists,  
defined ranges, and  
more



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help tests [C:\Users\Shayn\PycharmProjects\tests] - For loops.py - PyCharm
tests for loops.py
for loops.py
word = "test"
num = (1,2,3,4)

for i in word:
    print(i)

print()

for n in num:
    print(n)
```

Run: for loops x

```
t
e
s
t

1
2
3
4
```

Run | TODO | Terminal | Python Console | Event Log  
IDE and Plugin Updates: PyCharm is ready to update. (today 11:05) 13:5 CRLF UTF-8 4 spaces Python 3.7 (tests)



# Range vs. List in Loops

List iteration allows printing data.

A set range, defined by ***range()***, is also capable of iteration.

A variable is defined within the loop as the iterating item.

```
Michigan Python - Range.py
Range.py
1 fruit = ["Watermelon", "Banana", "Apple", "Pineapple", "Strawberry"]
2
3 for i in fruit:
4     print(i, end=" ")
5
6 print("\n")
7
8 for i in range(0,5):
9     print(i, end=" ")
10
11 print("\n")
12
13 for i in range(0, len(fruit)):
14     print(i, end=" ")

Run: Range
Watermelon Banana Apple Pineapple Strawberry

0 1 2 3 4

0 1 2 3 4
Process finished with exit code 0
```



For & While Loops

# Calculate Loop Size

The ***len()*** function returns the number of items of an object.

The ***range()*** function accepts an integer and returns the range of the object.

```
lesson.py
namelist = ["David", "John", "Cooper", "Nick"]
print(namelist, "The length is {}".format(len(namelist)))

colorlist = ["Blue", "Red", "Yellow"]

for i in range(8):
    print(i)
```

```
Run: lesson
['David', 'John', 'Cooper', 'Nick'] The length is 4
0
1
2
3
4
5
6
7
```



For & While Loops

# Lab PY-03-L1

Range Loop  
10–20 Min.



## Mission

Practice how to create a range with various inputs in a loop.

## Steps

- Request a range.
- Iterate the range.
- Display the iterated values.

## Environment & Tools

- Python 3
- PyCharm

## Related Files

- Lab document

For & While Loops

# Lab PY-03-L2

Loops in Nested Lists  
15–25 Min.



## Mission

Practice running nested loops.

## Steps

- Open a new project in PyCharm, and create a list called **Classroom**.
- Create a loop for classrooms.
- Create a loop for students.

## Environment & Tools

- Python 3
- PyCharm

## Related Files

- Lab document

# For & While Loops

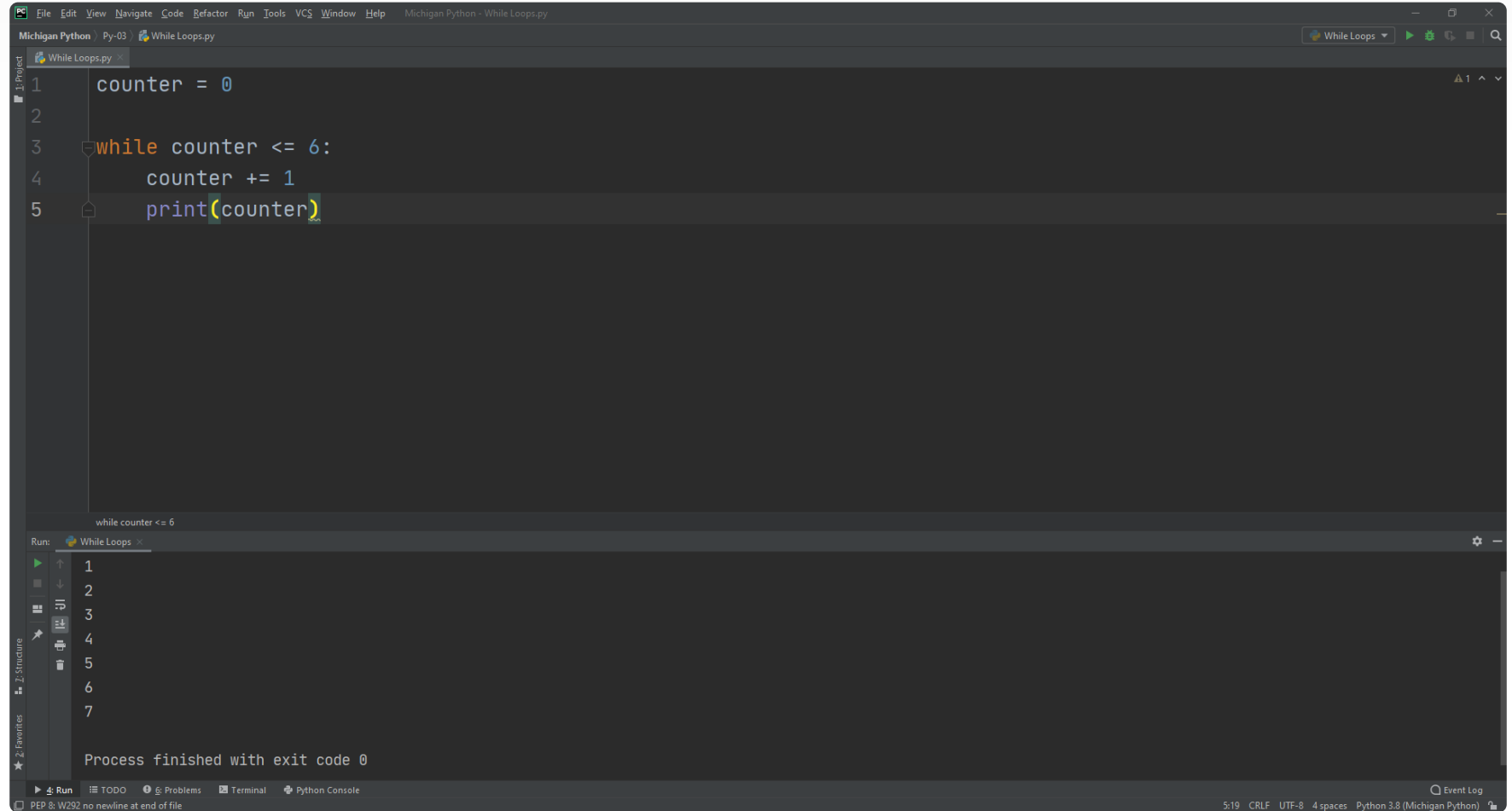
## *While* Loops



Perform a block of code while a condition is true

If the condition is false, the **while** loop will not be executed.

**While** loops do not have iteration capability like **for** loops.



```
1 counter = 0
2
3 while counter <= 6:
4     counter += 1
5     print(counter)
```

The screenshot shows a code editor window titled "Michigan Python - While Loops.py". The code in the editor is a while loop that increments a counter from 0 to 6 and prints the value. The output of the program is shown in the "Run" panel at the bottom, displaying the numbers 1 through 7 on separate lines, followed by the message "Process finished with exit code 0".

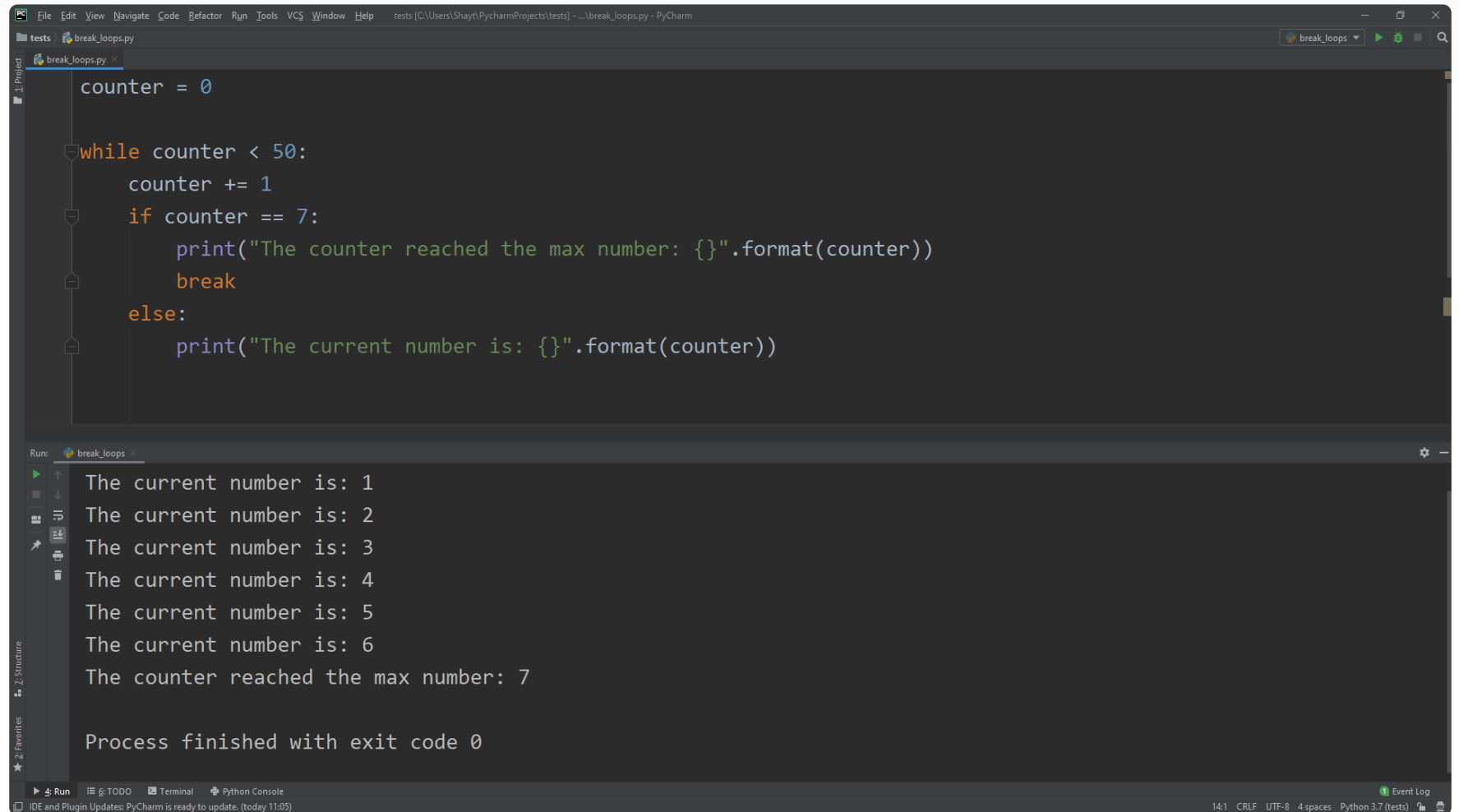


# For & While Loops *Break* Command

Both *for* and *while* loops can be interrupted.

The *break* command exits a loop.

Can be placed strategically for flow control



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help tests [C:\Users\Shayt\PycharmProjects\tests] - ...break_loops.py - PyCharm
break_loops.py
counter = 0


while counter < 50:
    counter += 1
    if counter == 7:
        print("The counter reached the max number: {}".format(counter))
        break
    else:
        print("The current number is: {}".format(counter))

Run: break_loops
The current number is: 1
The current number is: 2
The current number is: 3
The current number is: 4
The current number is: 5
The current number is: 6
The counter reached the max number: 7

Process finished with exit code 0
```

# For & While Loops

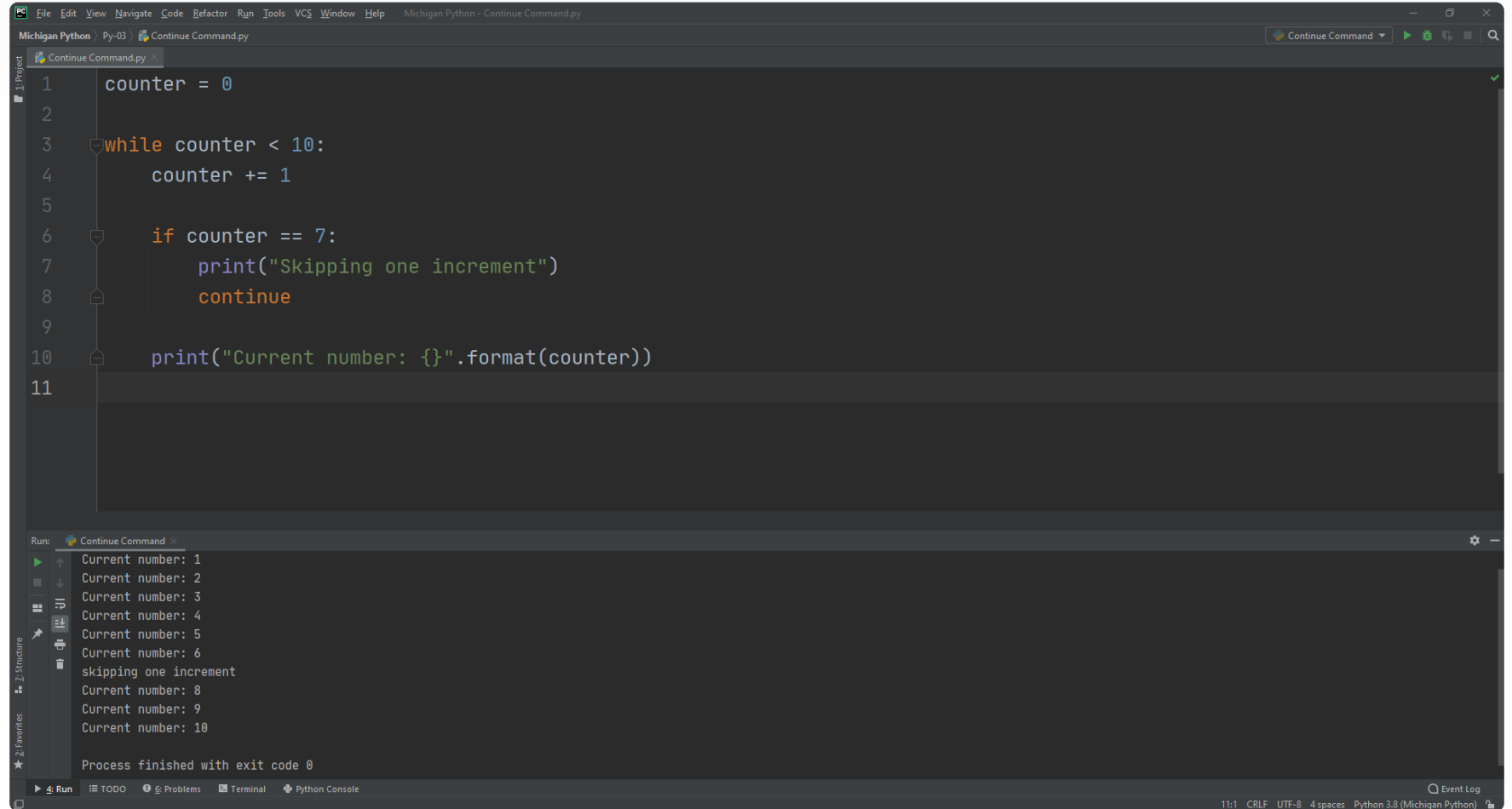
# *Continue* Command



Can be used in both *for* and *while* loops

The *continue* command skips an iteration.

Behaves as if the code reached the end of the block



```
1 counter = 0
2
3 while counter < 10:
4     counter += 1
5
6     if counter == 7:
7         print("Skipping one increment")
8         continue
9
10    print("Current number: {}".format(counter))
11
```

Run: Continue Command

```
Current number: 1
Current number: 2
Current number: 3
Current number: 4
Current number: 5
Current number: 6
skipping one increment
Current number: 8
Current number: 9
Current number: 10
Process finished with exit code 0
```

# For & While Loops

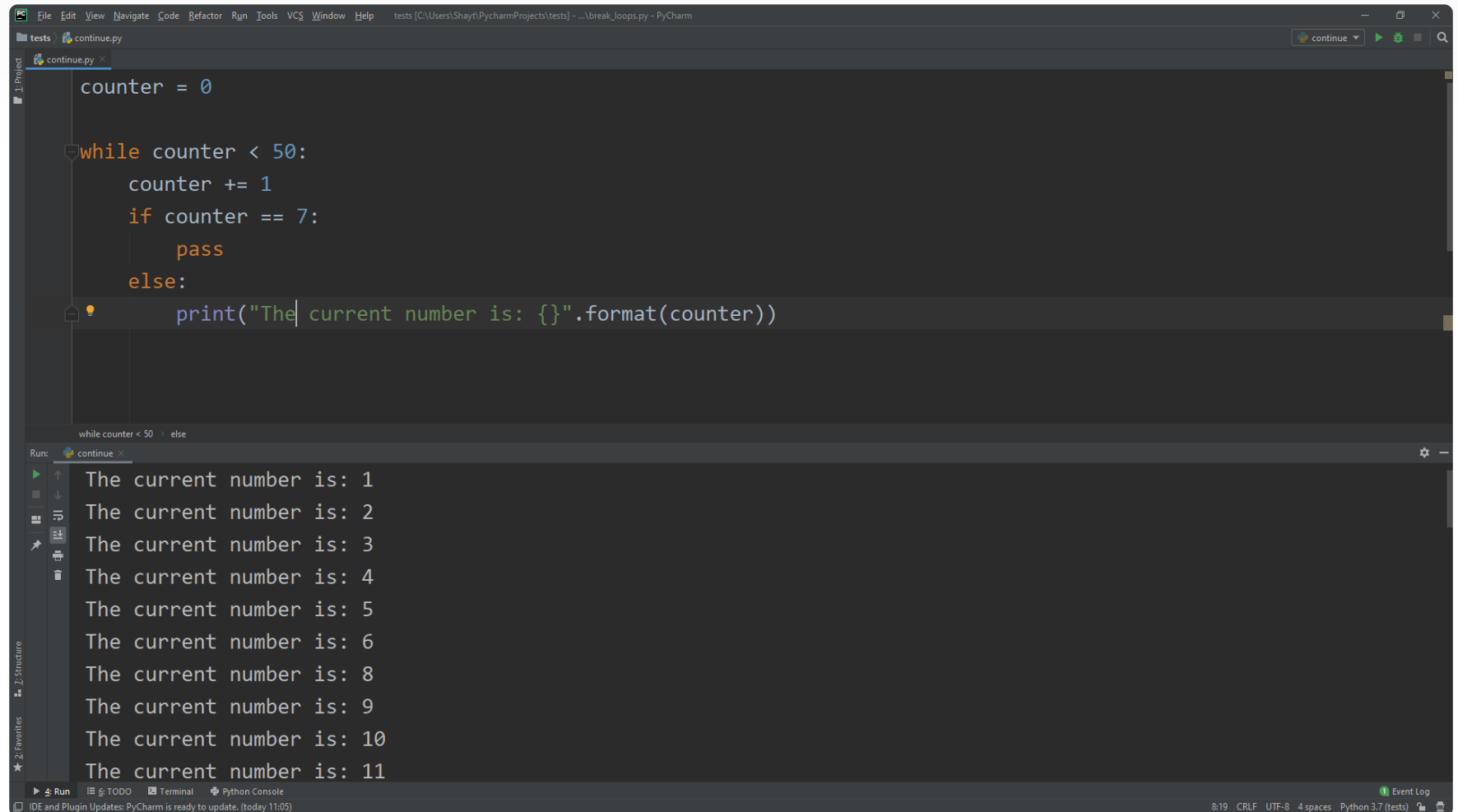
# *Pass* Command



***Pass*** statements are used as empty executions.

This is useful for when we don't want to execute any code.

***Pass*** can be used as a placeholder for future code.



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help tests [C:\Users\Shayt\PycharmProjects\tests] - ...break_loops.py - PyCharm
tests continue.py
continue.py
counter = 0
while counter < 50:
    counter += 1
    if counter == 7:
        pass
    else:
        print("The current number is: {}".format(counter))

Run: continue
The current number is: 1
The current number is: 2
The current number is: 3
The current number is: 4
The current number is: 5
The current number is: 6
The current number is: 8
The current number is: 9
The current number is: 10
The current number is: 11
```

The screenshot shows a PyCharm IDE window with a file named 'continue.py'. The code in the editor is a while loop that increments a counter from 0 to 50. Inside the loop, there is an if statement that checks if the counter is equal to 7. If it is, the code executes a 'pass' statement, which is highlighted in orange. Otherwise, it prints the current number. The output of the program is shown in the Run console at the bottom, displaying the numbers 1 through 11, with the number 7 being skipped.



For & While Loops

# *Pass vs. Continue*

***Pass*** means move on to the remaining code or loop body.

***Continue*** forces the loop to begin the next iteration.

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help tests [C:\Users\Shayt\PycharmProjects\tests] - ...mashu.py - PyCharm
tests mashu.py
mashu.py
word = ("test")
for x in word:
    if x == "s":
        print("Pass execute")
        pass
    print(x)
print()

for x in word:
    if x == "s":
        print("Continue execute")
        continue
    print(x)
```

Run: mashu x

```
t
e
Pass execute
s
t

t
e
Continue execute
t
```

IDE and Plugin Updates: PyCharm is ready to update. (today 11:05) 18:1 CRLF UTF-8 4 spaces Python 3.7 (tests) Event Log



Loops

---

# Loops & Conditions





Loops & Conditions

# Mixing Conditions and Loops

Conditions have a major role in a loop's flow control.

They provide greater flexibility for loops.

They also ensure more efficient conditional checks and help build a better code writing structure.

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help tests [C:\Users\Shayt\PycharmProjects\tests] - ...mix_conditions.py - PyCharm
tests mix_conditions.py
mix_conditions.py
numlist = [10, 22, 44, 30]
for i in numlist:
    if numlist[0] > 6 and numlist[3] > 40:
        print("Both numbers are bigger")
    elif numlist[1] == 22 or numlist[2] == 60:
        print("One of the numbers is equal")
        break
    else:
        print("None of this is true")
```

Run: mix\_conditions

```
One of the numbers is equal
Process finished with exit code 0
```

PEP 8: blank line at end of file 1:116 UTF-8 4 spaces Python 3.7 (tests) Event Log

# Loops & Conditions

## Infinite Loops



A loop becomes infinite if a condition is never *false*.

To avoid an infinite loop, the loop must verify the existence of a false condition.

```
num1 = 1
while num1 == 1:
    user = input("Please enter a number: ")
    print("You have entered the number {}".format(user))

print("Good Bye!")
```

You have entered the number 11  
Please enter a number: 44  
You have entered the number 44  
Please enter a number: 66  
You have entered the number 66  
Please enter a number: 88  
You have entered the number 88  
Please enter a number: 46  
You have entered the number 46



Loops & Conditions

# Nested Loops and Lists

The ***append()*** function can be used to define a list within another list.

***append()*** adds a value to an existing variable.

Nested lists can be used to arrange data in hierarchical structures.

```
numlist = []

for i in range(5):
    numlist.append([])
    for b in range(5):
        numlist[i].append(b)
print(numlist)
```

```
[[0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4]]

Process finished with exit code 0
```

For & While Loops

# Lab PY-03-L3

Loops with Conditions  
10–20 Min.



## Mission

Practice executing code with a ***while*** loop and conditions.

## Steps

- Create a counting mechanism.
- Locate a specific value when counting.

## Environment & Tools

- Python 3
- PyCharm

## Related Files

- Lab document

For & While Loops

# Lab PY-03-L4

User Dictionary  
15–20 Min.



## Mission

Work with input from users to control program flow and insert and retrieve the desired output.

## Steps

- Create an empty dictionary and two variables.
- Identify the service.
- Print the final dictionary if the user breaks the loop.

## Environment & Tools

- Python 3
- PyCharm

## Related Files

- Lab document

# Lab PY-03-L5

While and Conditions  
30–50 Min.



## Mission

Implement conditions to control infinite ***while*** loops.

## Steps

- Create a list of groceries with an available budget.
- Iterate the list for budget calculation.
- Simulate grocery shopping.

## Environment & Tools

- Python 3
- PyCharm

## Related Files

- Lab document





Thank You

---

Questions?